

# quanteda

September 22, 2014

**Type** Package

**Title** Quantitative Analysis of Textual Data

**Version** 0.4

**Date** 2014-09-17

**Author** Ken Benoit and Paul Nulty

**Maintainer** Ken Benoit <kbenoit@lse.ac.uk> and Paul Nulty <p.nulty@lse.ac.uk>

**Description** A library for the quantitative analysis of textual data with R

**Encoding** UTF-8

**License** GPL-3

**Requires** wordcloud, SnowballC, proxy

**Suggests** austin,entropy,jsonlite,openNLP,RJSONIO,RCurl,twitteR,XML

**URL** <http://github.com/kbenoit/quanteda>

**LazyData** TRUE

## R topics documented:

bigrams . . . . .	2
clean . . . . .	3
collocations . . . . .	4
corpus . . . . .	5
countSyllables . . . . .	6
describeTexts . . . . .	7
dfm . . . . .	8
dfm2ldaformat . . . . .	9
dfm2tmformat . . . . .	10
directory . . . . .	11
docnames . . . . .	11
docvars . . . . .	12
features.dfm . . . . .	13
flatten.dictionary . . . . .	13
getRootFileNames . . . . .	14

getTextDir . . . . .	15
getTextDirGui . . . . .	15
getTextFiles . . . . .	16
inaugCorpus . . . . .	17
kwic . . . . .	17
language . . . . .	18
likelihood.test . . . . .	19
metacorporus . . . . .	19
metadoc . . . . .	20
ndoc . . . . .	21
ngrams . . . . .	21
plot.dfm . . . . .	22
preprocess . . . . .	23
quanteda . . . . .	24
readWStatDict . . . . .	24
segment . . . . .	25
settings . . . . .	26
sort.dfm . . . . .	27
stopwords . . . . .	27
stopwordsGet . . . . .	28
stopwordsRemove . . . . .	28
subset.corpus . . . . .	29
summary.corpus . . . . .	30
syllableCounts . . . . .	30
texts . . . . .	31
tf . . . . .	31
tfidf.dfm . . . . .	32
tokenize . . . . .	33
topfeatures . . . . .	34
trimdfm . . . . .	34
twitterSearch . . . . .	35
twitterStreamer . . . . .	35
twitterTerms . . . . .	36
uk2010immig . . . . .	36

<b>Index</b>	<b>37</b>
--------------	-----------

---

bigrams	<i>Create bigrams</i>
---------	-----------------------

---

## Description

Create bigrams

## Usage

```
bigrams(text, window = 1, concatenator = "_", include.unigrams = FALSE,
...)
```

**Arguments**

text	character vector containing the texts from which bigrams will be constructed
window	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words. This is only available for bigrams, not for <a href="#">ngram</a> .
concatenator	character for combining words, default is _ (underscore) character
include.unigrams	if TRUE, return unigrams as well
...	additional arguments passed to <a href="#">tokenize</a>

**Value**

a character vector of bigrams

**Author(s)**

Ken Benoit and Kohei Watanabe

**Examples**

```
bigrams("The quick brown fox jumped over the lazy dog.")
bigrams(c("The quick brown fox", "jumped over the lazy dog."))
bigrams(c("The quick brown fox", "jumped over the lazy dog."), window=2)
```

---

clean	<i>simple cleaning of text before processing</i>
-------	--

---

**Description**

clean removes punctuation and digits from text, using the regex character classes for punctuation and digits. clean uses the standard R function tolower to convert the text to lower case. Each of these steps is optional, but switched on by default, so for example, to remove punctuation and convert to lower, but keep digits, the command would be: clean(mytexts, removeDigits=FALSE)

**Usage**

```
clean(x, ...)

## S3 method for class character
clean(x, removeDigits = TRUE, removePunct = TRUE,
      lower = TRUE, ...)

## S3 method for class corpus
clean(x, removeDigits = TRUE, removePunct = TRUE,
      lower = TRUE, ...)
```

**Arguments**

x	The object to be cleaned. Can be either a character vector or a corpus object. If x is a corpus, clean returns a copy of the x with the texts cleaned.
removeDigits	remove numbers if TRUE
removePunct	remove punctuation if TRUE
lower	convert text to lower case TRUE

**Value**

A character vector equal in length to the original texts, after cleaning.

**Examples**

```
clean("This is 1 sentence with 1.9 numbers in it, and one comma.", removeDigits=FALSE)
clean("This is 1 sentence with 1.9 numbers in it, and one comma.", lower=FALSE)

# for a vector of texts
clean(c("This is 1 sentence with 1.9 numbers in it, and one comma.",
        "€1.2 billion was spent on text analysis in 2014."))
```

---

collocations	<i>Detect collocations in a text</i>
--------------	--------------------------------------

---

**Description**

returns a list of collocations. Note: Currently works only for pairs (bigram collocations).

**Usage**

```
collocations(text = NULL, file = NULL, top = NA, distance = 2, n = 2,
             method = c("lr", "chi2", "mi"))
```

**Arguments**

text	a text or vector of texts
file	a filename containing a text
top	threshold number for number of collocations to be returned (in descending order of association value)
distance	distance between pairs of collocations
n	Only bigrams (n=2) implemented so far.
method	association measure for detecting collocations

**Value**

A list of collocations, their frequencies, and their test statistics

**Author(s)**

Kenneth Benoit

**Examples**

```
data(inaugCorpus)
collocations(texts(inaugCorpus)[1], top=50)
collocations(texts(inaugCorpus)[1], top=50, method="chi2")
```

corpus

*Constructor for corpus objects***Description**

Creates a corpus from a document source, such as character vector (of texts), or an object pointing to a source of texts such as a directory containing text files. Corpus-level meta-data can be specified at creation, containing (for example) citation information and notes.

**Usage**

```
corpus(x, ...)

## S3 method for class directory
corpus(x, enc = NULL, docnames = NULL,
       docvarsfrom = c("filenames", "headers"), docvarnames = NULL, sep = "_",
       source = NULL, notes = NULL, citation = NULL, ...)

## S3 method for class character
corpus(x, enc = NULL, docnames = NULL, docvars = NULL,
       source = NULL, notes = NULL, citation = NULL, ...)

is.corpus(x)
```

**Arguments**

x	A source of texts to form the documents in the corpus. This can be a filepath to a directory containing text documents (see <a href="#">directory</a> ), or a character vector of texts.
docnames	Names to be assigned to the texts, defaults to the names of the character vector (if any), otherwise assigns "text1", "text2", etc.
docvarsfrom	Argument to specify where docvars are to be taken, from parsing the filenames ( <a href="#">filenames</a> ) separated by sep or from meta-data embedded in the text file header (headers).
docvarnames	Character vector of variable names for docvars
sep	Separator if <a href="#">docvar</a> names are taken from the filenames.
source	A string specifying the source of the texts, used for referencing.
notes	A string containing notes about who created the text, warnings, To Dos, etc.
docvars	A data frame of attributes that is associated with each text.

**Value**

A corpus class object containing the original texts, document-level variables, document-level meta-data, corpus-level metadata, and default settings for subsequent processing of the corpus. A corpus consists of a list of elements described below, although these should only be accessed through accessor and replacement functions, not directly (since the internals may be subject to change). The structure of a corpus classed list object is:

<code>\$documents</code>	A data frame containing the document level information, consisting of <a href="#">texts</a> , user-named <a href="#">docvars</a> variables describing attributes of the documents, and <code>metadoc</code> document-level metadata whose names begin with an underscore character, such as <a href="#">_language</a> .
<code>\$metadata</code>	A named list set of corpus-level meta-data, including source and created (both generated automatically unless assigned), notes, and citation.
<code>\$settings</code>	Settings for the corpus which record options that govern the subsequent processing of the corpus when it is converted into a document-feature matrix ( <a href="#">dfm</a> ). See <a href="#">settings</a> .
<code>\$tokens</code>	An indexed list of tokens and types tabulated by document, including information on positions. Not yet fully implemented.

`is.corpus` returns TRUE if the object is a corpus

### See Also

[docvars](#), [metadoc](#), [metacorporus](#), [language](#), [encoding](#), [settings](#), [texts](#)

### Examples

```
## Not run:
# import texts from a directory of files
corpus(directory("~/Dropbox/QUANTESS/corpora/ukManRenamed"),
        enc="UTF-8",
        source="Kens UK manifesto archive")

# choose a directory using a GUI
corpus(directory())
## End(Not run)
#
# create a corpus from texts
corpus(inaugTexts)

# create a corpus from texts and assign meta-data and document variables
uk2010immigCorpus <- corpus(uk2010immig,
                           docvars=data.frame(party=names(uk2010immig)),
                           enc="UTF-8")
```

---

<code>countSyllables</code>	<i>Returns a count of the number of syllables in the input This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.</i>
-----------------------------	--

---

### Description

Returns a count of the number of syllables in the input This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.

**Usage**

```
countSyllables(sourceText)
```

**Arguments**

sourceText      Character vector of texts whose syllables will be counted

**Details**

This only works for English.

**Value**

numeric Named vector of counts of the number of syllables for each element of sourceText. When a word is not available in the lookup table, its syllables are estimated by counting the number of (English) vowels in the word.

**Examples**

```
countSyllables("This is an example sentence.")
myTexts <- c("Text one.", "Superduper text number two.", "One more for the road.")
names(myTexts) <- paste("myText", 1:3, sep="")
countSyllables(myTexts)
```

---

describeTexts	<i>print a summary of texts Prints to the console a description of the texts, including number of types, tokens, and sentences</i>
---------------	--

---

**Description**

print a summary of texts Prints to the console a description of the texts, including number of types, tokens, and sentences

**Usage**

```
describeTexts(txts, verbose = TRUE)
```

**Arguments**

txts              The texts to be described

verbose           Default is TRUE. Set to false to suppress output messages

**Examples**

```
describeTexts(c("testing this text", "and this one"))
describeTexts(uk2010immig)
```

dfm

*Create a document-feature matrix from a corpus object***Description**

returns a document by feature matrix compatible with `austin`. A typical usage would be to produce a word-frequency matrix where the cells are counts of words by document.

**Usage**

```
dfm(x, ...)

## S3 method for class corpus
dfm(x, feature = c("word"), stem = FALSE,
    stopwords = NULL, bigram = FALSE, groups = NULL, verbose = TRUE,
    dictionary = NULL, dictionary_regex = FALSE, clean = TRUE,
    removeDigits = TRUE, removePunct = TRUE, lower = TRUE, addto = NULL,
    ...)

## S3 method for class character
dfm(x, feature = c("word"), stem = FALSE,
    stopwords = NULL, bigram = FALSE, verbose = TRUE, dictionary = NULL,
    dictionary_regex = FALSE, clean = TRUE, removeDigits = TRUE,
    removePunct = TRUE, lower = TRUE, addto = NULL, ...)

is.dfm(x)
```

**Arguments**

<code>x</code>	Corpus or character vector from which to generate the document-feature matrix
<code>feature</code>	Feature to count (e.g. words)
<code>stem</code>	Stem the words
<code>stopwords</code>	A character vector of stopwords that will be removed from the text when constructing the <a href="#">dfm</a> . If <code>NULL</code> (default) then no stopwords will be applied. If <code>"TRUE"</code> then it currently defaults to <a href="#">stopwords</a> .
<code>groups</code>	Grouping variable for aggregating documents
<code>verbose</code>	Get info to screen on the progress
<code>dictionary</code>	A list of character vector dictionary entries, including regular expressions (see examples)
<code>dictionary_regex</code>	<code>TRUE</code> means the dictionary is already in regular expression format, otherwise it will be converted from "wildcard" format
<code>addto</code>	<code>NULL</code> by default, but if an existing <code>dfm</code> object is specified, then the new <code>dfm</code> will be added to the one named. If both <a href="#">dfm</a> 's are built from dictionaries, the combined <code>dfm</code> will have its <code>Non_Dictionary</code> total adjusted.

**Details**

`is.dfm` returns `TRUE` if and only if its argument is a [dfm](#).



**Value**

A matrix object with row names equal to the document names and column names equal to the feature labels. This matrix has `names(dimnames) = c("docs", "words")` to make it conformable to an [wfm](#) object.

**Author(s)**

Kenneth Benoit

**Examples**

```
data(inaugCorpus)
wfm <- dfm(inaugCorpus)

## by president, after 1960
wfmByPresfrom1900 <- dfm(subset(inaugCorpus, Year>1900), groups="President")
docnames(wfmByPresfrom1900)

## with dictionaries
data(inaugCorpus)
mycorpus <- subset(inaugCorpus, Year>1900)
mydict <- list(christmas=c("Christmas", "Santa", "holiday"),
               opposition=c("Opposition", "reject", "notincorpus"),
               taxing="taxing",
               taxation="taxation",
               taxregex="tax*")
dictDfm <- dfm(mycorpus, dictionary=mydict)
dictDfm

## removing stopwords
testText <- "The quick brown fox named Seamus jumps over the lazy dog also named Seamus, with
             the newspaper from a a boy named Seamus, in his mouth."
testCorpus <- corpus(testText)
settings(testCorpus, "stopwords")
dfm(testCorpus, stopwords=TRUE)
```

---

dfm2ldaformat

---

Convert a [dfm](#) into the format needed by [lda](#)


---

**Description**

Convert a quanteda [dfm](#) object into the indexed format required by the topic modelling package [lda](#).

**Usage**

```
dfm2ldaformat(d)
```

**Arguments**

`d`                      A [dfm](#) object

**Value**

A list with components "documents" and "vocab" as needed by [lda.collapsed.gibbs.sampler](#)

## Examples

```
mycorpus <- subset(inaugCorpus, Year>1970)
d <- dfm(mycorpus, stopwords=TRUE)
d <- trimdfm(d, minCount=5, minDoc=3)
td <- dfm2ldaformat(d)
if (require(lda)) {
  tmodel.lda <- lda.collapsed.gibbs.sampler(documents=td$documents,
                                           K=10,
                                           vocab=td$vocab,
                                           num.iterations=50, alpha=0.1, eta=0.1)
  top.topic.words(tmodel.lda$topics, 10, by.score=TRUE) # top five words in each topic
}
```

---

dfm2tmformat

---

Convert a *dfm* into a **tm** *DocumentTermMatrix*


---

## Description

**tm** represents sparse document-feature matrixes in the [simple triplet matrix](#) format of the package **slam**. This function converts a *dfm* into a [DocumentTermMatrix](#), enabling a *dfm* to be used with other packages that expect this format, such as **topicmodels**.

## Usage

```
dfm2tmformat(d, weighting = weightTf, ...)
```

## Arguments

d	A <a href="#">dfm</a> object
weighting	weight function arguments passed to <code>as.TermDocumentMatrix</code> , defaults to term frequency (see <a href="#">as.DocumentTermMatrix</a> for a list of options, such as <code>tf-idf</code> ).

## Value

A simple triplet matrix of class [as.DocumentTermMatrix](#)

## Examples

```
mycorpus <- subset(inaugCorpus, Year>1970)
d <- trimdfm(dfm(mycorpus), minCount=5, minDoc=3)
dim(d)
td <- dfm2tmformat(d)
length(td$v)
if (require(topicmodels)) (tmodel.lda <- LDA(td, control = list(alpha = 0.1), k = 5))
```

---

directory	<i>Function to declare a connection to a directory (containing files)</i>
-----------	---

---

### Description

Function to declare a connection to a directory, although unlike [file](#) it does not require closing. If the directory does not exist, the function will return an error.

### Usage

```
directory(path = NULL)
```

### Arguments

path	String describing the full path of the directory or NULL to use a GUI to choose a directory from disk
------	---

### Examples

```
## Not run:
# name a directory of files
mydir <- directory("~/Dropbox/QUANTESS/corpora/ukManRenamed")
corpus(mydir)

# choose a directory using a GUI
corpus(directory())
## End(Not run)
```

---

docnames	<i>extract document names</i>
----------	-------------------------------

---

### Description

Extract the document names from a corpus or a document-feature matrix. Document names are the rownames of the [documents](#) data.frame in a corpus, or the rownames of the [dfm](#) object for a dfm. of the [dfm](#) object.

docnames queries the document names of a corpus or a dfm

docnames <- assigns new values to the document names of a corpus. (Does not work for dfm objects, whose document names are fixed,)

### Usage

```
docnames(x)

## S3 method for class corpus
docnames(x)

docnames(x) <- value

## S3 method for class dfm
docnames(x)
```

**Value**

docnames returns a character vector of the document names

docnames<- assigns a character vector of the document names in a corpus

**Examples**

```
# query the document names of the inaugural speech corpus
docnames(inaugCorpus) <- paste("Speech", 1:ndoc(inaugCorpus), sep="")

# reassign the document names of the inaugural speech corpus
docnames(inaugCorpus) <- paste("Speech", 1:ndoc(inaugCorpus), sep="")
#
# query the document names of a dfm
docnames(dfm(inaugTexts[1:5]))
```

---

docvars

*get or set for document-level variables*

---

**Description**

Get or set variables for the documents in a corpus

**Usage**

```
docvars(x)
```

```
docvars(x, field) <- value
```

**Arguments**

x	corpus whose document-level variables will be read or set
field	string containing the document-level variable name

**Value**

docvars returns a data.frame of the document-level variables

docvars<- assigns value to the named field

**Examples**

```
head(docvars(inaugCorpus))
docvars(inaugCorpus, "President") <- paste("prez", 1:ndoc(inaugCorpus), sep="")
head(docvars(inaugCorpus))
```

---

features.dfm	<i>extract the feature labels from a <a href="#">dfm</a></i>
--------------	--

---

**Description**

Extract the features from a document-feature matrix, which are stored as the column names of the [dfm](#) object.

**Usage**

```
## S3 method for class dfm
features(x)
```

**Value**

Character vector of the features

**Examples**

```
features(dfm(inaugTexts))[1:50] # first 50 features (alphabetically sorted)
```

---

flatten.dictionary	<i>Flatten a hierarchical dictionary into a list of character vectors</i>
--------------------	---

---

**Description**

Converts a hierarchical dictionary (a named list of named lists, ending in character vectors at the lowest level) into a flat list of character vectors. Works like `unlist(dictionary, recursive=TRUE)` except that the recursion does not go to the bottom level.

**Usage**

```
flatten.dictionary(elms, parent = "", dict = list())
```

**Arguments**

elms	list to be flattened
parent	parent list name, gets built up through recursion in the same way that <code>unlist(dictionary, recursive=TRUE)</code> works
dict	the bottom list of dictionary entries ("synonyms") passed up from recursive calls

**Details**

Called by `dfm()`

**Value**

A dictionary flattened down one level further than the one passed

**Author(s)**

Kohei Watanabe

**Examples**

```
dictPopulismEN <-
  list(populism=c("elit*", "consensus*", "undemocratic*", "referend*",
                 "corrupt*", "propagand", "politici*", "*deceit*",
                 "*deceiv*", "*betray*", "shame*", "scandal*", "truth*",
                 "dishonest*", "establishm*", "ruling*"))
flatten.dictionary(dictPopulismEN)

hdict <- list(level1a = list(level1a1 = c("l1a11", "l1a12"),
                             level1a2 = c("l1a21", "l1a22")),
              level1b = list(level1b1 = c("l1b11", "l1b12"),
                             level1b2 = c("l1b21", "l1b22", "l1b23")),
              level1c = list(level1c1a = list(level1c1a1 = c("lowest1", "lowest2")),
                             level1c1b = list(level1c1b1 = c("lowestalone"))))
flatten.dictionary(hdict)
```

getRootFileNames

*Truncate absolute filepaths to root filenames***Description**

This function takes an absolute filepath and returns just the document name

**Usage**

```
getRootFileNames(longFilenames)
```

**Arguments**

longFilenames Absolute filenames including a full path with directory

**Value**

character vector of filenames withouth directory path

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:
getRootFileNames(/home/paul/documents/libdem09.txt)

## End(Not run)
```

---

getTextDir	<i>loads all text files from a given directory</i>
------------	--

---

**Description**

given a directory name, get a list of all files in that directory and load them into a character vector using getTextFiles

**Usage**

```
getTextDir(dirname, enc = "detect", pattern = "\\..txt$")
```

**Arguments**

dirname	A directory path
---------	------------------

**Value**

character vector of texts read from disk

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:  
getTextDir(/home/paul/documents/)  
  
## End(Not run)
```

---

getTextDirGui	<i>provides a gui interface to choose a gui to load texts from</i>
---------------	--

---

**Description**

launches a GUI to allow the user to choose a directory from which to load all files.

**Usage**

```
getTextDirGui()
```

**Value**

character vector of texts read from disk

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:
getTextFiles(/home/paul/documents/libdem09.txt)

## End(Not run)
```

---

getTextFiles	<i>load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts</i>
--------------	--

---

**Description**

load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts

**Usage**

```
getTextFiles(filenamees, textnames = NULL, enc = "unknown",
  verbose = FALSE)
```

**Arguments**

filenamees	a vector of paths to text files
textnames	names to assign to the texts
verbose	If TRUE, print out names of files being read. Default is FALSE

**Value**

character vector of texts read from disk

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:
getTextFiles(/home/paul/documents/libdem09.txt)

## End(Not run)
```



---

inaugCorpus

*A corpus of US presidential inaugural addresses from 1789-2013*


---

### Description

inaugCorpus is the [quanteda](#) corpus object of US presidents' inaugural addresses since 1789. Document variables contain the year of the address and the last name of the president.

inaugTexts is the character vector of US presidential inauguration speeches

### References

<https://archive.org/details/Inaugural-Address-Corpus-1789-2009> and <http://www.presidency.ucsb.edu/inaugurals.php>.

### Examples

```
# some operations on the inaugural corpus
data(inaugCorpus)
summary(inaugCorpus)
head(docvars(inaugCorpus), 10)
# working with the character vector only
data(inaugTexts)
str(inaugTexts)
head(docvars(inaugCorpus), 10)
mycorpus <- corpus(inaugTexts)
```

---

kwic

*List key words in context from a text or a corpus of texts.*


---

### Description

For a text or a collection of texts (in a [quanteda](#) corpus object), return a list of a keyword supplied by the user in its immediate context, identifying the source text and the word index number within the source text. (Not the line number, since the text may or may not be segmented using end-of-line delimiters.)

### Usage

```
kwic(x, word, window = 5, regex = TRUE)

## S3 method for class character
kwic(x, word, window = 5, regex = TRUE)

## S3 method for class corpus
kwic(x, word, window = 5, regex = TRUE)
```

**Arguments**

x	A text character scalar or a quanteda corpus. (Currently does not support character vectors.)
word	A keyword chosen by the user.
window	The number of context words to be displayed around the keyword.
regex	If TRUE (default), then "word" is a regular expression, otherwise only match the whole word. Note that if regex=TRUE and no special regular expression characters are used in the search query, then the concordance will include all words in which the search term appears, and not just when it appears as an entire word. (For instance, searching for the word "key" will also return "whiskey".)
texts	a vector of texts
corp	a quanteda corpus object

**Value**

A data frame with the context before (preword), the keyword in its original format (word, preserving case and attached punctuation), and the context after (postword). The rows of the dataframe will be named with the word index position, or the text name and the index position for a corpus object.

**Author(s)**

Kenneth Benoit and Paul Nulty

**Examples**

```
kwic(inaugTexts, "terror")
kwic(inaugTexts, "terror", regex=FALSE) # returns only whole word, without trailing punctuation
data(iebudgets)
kwic(subset(iebudgets, year==2010), "Christmas", window=4) # on a corpus
```

---

language

*get or set the language of corpus documents*

---

**Description**

Get or set the `_language` document-level metadata field in a corpus. Same as

**Usage**

```
language(corp)
```

---

likelihood.test	<i>likelihood test for 2x2 tables</i>
-----------------	---------------------------------------

---

**Description**

returns a list of values

**Usage**

```
likelihood.test(x)
```

**Arguments**

`x` a contingency table or matrix object

**Value**

A list of return values

**Author(s)**

Kenneth Benoit

---

metacorporus	<i>get or set corpus metadata</i>
--------------	-----------------------------------

---

**Description**

Get or set the corpus-level metadata in a quanteda corpus object.

**Usage**

```
metacorporus(corp, field = NULL)
```

```
metacorporus(corp, field) <- value
```

**Arguments**

`corp` A quanteda corpus object

`field` Metadata field name(s). If NULL (default), return all metadata names.

**Value**

For `metacorporus`, a list of the metadata fields in the corpus. If a list is not what you wanted, you can wrap the results in [unlist](#), but this will remove any metadata field that is set to NULL.

For `metacorporus <-`, the corpus with the updated metadata.

**Examples**

```
metacorp(inaugCorpus)
metacorp(inaugCorpus, "source")
metacorp(inaugCorpus, "citation") <- "Presidential Speeches Online Project (2014)."
```

```
metacorp(inaugCorpus, "citation")
```

---

metadoc

*get or set document-level meta-data*


---

**Description**

Get or set the document-level meta-data, including reserved fields for language and corpus.

**Usage**

```
metadoc(corp, field = NULL)
```

**Arguments**

corp                      A quanteda corpus object

**Value**

For texts, a character vector of the texts in the corpus.

For texts <-, the corpus with the updated texts.

**Note**

Document-level meta-data names are preceded by an underscore character, such as `_encoding`, but when named in in the `field` argument, do *not* need the underscore character.

**Examples**

```
mycorp <- subset(inaugCorpus, Year>1990)
summary(mycorp, showmeta=TRUE)
metadoc(mycorp, "encoding") <- "UTF-8"
metadoc(mycorp)
metadoc(mycorp, "language") <- "english"
summary(mycorp, showmeta=TRUE)
```

---

ndoc	<i>get the number of documents</i>
------	------------------------------------

---

### Description

Returns the number of documents in a corpus objects

### Usage

```
## S3 method for class corpus
ndoc(x, ...)

## S3 method for class dfm
ndoc(x)
```

### Arguments

x a corpus or dfm object

### Value

an integer (count) of the number of documents in the corpus or dfm

### Examples

```
ndoc(inaugCorpus)
ndoc(dfm(inaugCorpus))
```

---

ngrams	<i>Create ngrams</i>
--------	----------------------

---

### Description

Create a set of ngrams (words in sequence) from text(s) in a character vector

### Usage

```
ngrams(text, n = 2, concatenator = "_", include.all = FALSE, ...)
```

### Arguments

text	character vector containing the texts from which ngrams will be extracted
n	the number of tokens to concatenate. Default is 2 for bigrams.
concatenator	character for combining words, default is _ (underscore) character
include.all	if TRUE, add n-1...1 grams to the returned list
...	additional arguments passed to <a href="#">tokenize</a>
window	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words.

**Value**

a list of character vectors of ngrams, one list element per text

**Author(s)**

Ken Benoit, Kohei Watanabe, Paul Nulty

**Examples**

```
ngrams("The quick brown fox jumped over the lazy dog.", n=2)
identical(ngrams("The quick brown fox jumped over the lazy dog.", n=2),
          bigrams("The quick brown fox jumped over the lazy dog.", n=2))
ngrams("The quick brown fox jumped over the lazy dog.", n=3)
ngrams("The quick brown fox jumped over the lazy dog.", n=3, concatenator="~")
ngrams("The quick brown fox jumped over the lazy dog.", n=3, include.all=TRUE)
```

---

plot.dfm

*plot features as a wordcloud*

---

**Description**

The default plot method for a [dfm](#) object. Produces a wordcloud plot for the features of the dfm, weighted by the total frequencies. To produce word cloud plots for specific documents, the only way currently to do this is to produce a dfm only from the documents whose features you want plotted.

**Usage**

```
## S3 method for class dfm
plot(x, ...)
```

**Arguments**

x	a dfm object
...	additional parameters to <a href="#">wordcloud</a> or to <a href="#">text</a> (and <a href="#">strheight</a> , <a href="#">strwidth</a> )

**See Also**

[wordcloud](#)

**Examples**

```
# plot the features (without stopwords) from Obamas two inaugural addresses
mydfm <- dfm(subset(inaugCorpus, President=="Obama"), verbose=FALSE, stopwords=TRUE)
plot(mydfm)

# plot only Lincolns inaugural address
plot(dfm(subset(inaugCorpus, President=="Lincoln"), verbose=FALSE, stopwords=TRUE))

# plot in colors with some additional options passed to wordcloud
plot(mydfm, random.color=TRUE, rot.per=.25, colors=sample(colors()[2:128], 5))
```

---

preprocess	<i>preprocess the tokens in a corpus</i>
------------	--

---

### Description

Applies pre-processing rules to the text and compiles a frequency table of features (word types) including counts of types, tokens, sentences, and paragraphs.

### Usage

```
preprocess(corp)
```

### Arguments

corp	Corpus to be preprocessed
------	---------------------------

### Value

no return but modifies the object in place by changing

tokens,	a list consisting of the following:
\$dfm	A <a href="#">dfm</a> document-feature matrix object created with <a href="#">settings</a> .
\$nwords	A vector of token counts for each document.
\$ntypes	A vector of type counts for each document.
\$nsents	A vector of sentence counts for each document.
\$nparagr	A vector of paragraph counts for each document.

### Note

This will eventually become an indexing function. At the moment it creates and saves a [dfm](#) in addition to some summary information compiled from this, in order to speed up subsequent processing. Unlike most R functions which return a value, this one changes the object passed to it. (And they say R can't pass by reference...)

### Examples

```
mycorpus <- corpus(uk2010immig)
mycorpus
preprocess(mycorpus)
mycorpus
mydfm <- dfm(mycorpus)
```

---

quanteda

*An R package for the quantitative analysis of textual data.*

---

### Description

A set of functions for creating and managing text corpora, extracting features from text corpora, and analyzing those features using quantitative methods.

### Author(s)

Ken Benoit and Paul Nulty

---

readWStatDict

*Import a Wordstat dictionary*

---

### Description

Make a flattened list from a hierarchical wordstat dictionary

### Usage

```
readWStatDict(path)
```

### Arguments

path                      path to the wordstat dictionary file (.cat)

### Value

a named list, where each the name of element is a bottom level category in the hierarchical wordstat dictionary. Each element is a list of the dictionary terms corresponding to that level.

### Author(s)

Kohei Watanabe

### Examples

```
\donrun{
path <- ~/Dropbox/QUANTESS/corpora/LaverGarry.cat
lgdict <- readWStatDict(path)
}
```



---

segment	<i>segment texts into component elements</i>
---------	--

---

## Description

Segment text(s) into tokens, sentences, paragraphs, or other sections. `segment` works on a character vector or corpus object, and allows the delimiters to be defined. See details.

## Usage

```
segment(x, ...)

## S3 method for class character
segment(x, what = c("tokens", "sentences", "paragraphs",
  "other"), delimiter = ifelse(what == "tokens", " ", ifelse(what ==
  "sentences", "[.!?;:]", "\\n{2}")), ...)

## S3 method for class corpus
segment(x, what = c("tokens", "sentences", "paragraphs",
  "other"), delimiter = ifelse(what == "tokens", " ", ifelse(what ==
  "sentences", "[.!?;:]", "\\n{2}")), ...)
```

## Arguments

<code>...</code>	additional arguments to be passed to <a href="#">clean</a>
<code>what</code>	defines the component to define the segmentation unit. Current options are tokens, sentences, paragraphs, and other. Segmenting on other allows segmentation of a text on any user-defined value, and must be accompanied by the <code>delimiter</code> argument.
<code>delimiter</code>	delimiter defined as a <a href="#">regex</a> for segmentation. Each type has its own default, except other, which requires a value to be specified.

## Details

Tokens are delimited by whitespace. For sentences, the delimiter can be defined by the user. The default for sentences includes ., !, ?, plus ; and :.

For paragraphs, the default is two carriage returns, although this could be changed to a single carriage return by changing the value of `delimiter` to `"\\n{1}"` which is the R version of the [regex](#) for one newline character. (You might need this if the document was created in a word processor, for instance, and the lines were wrapped in the window rather than being hard-wrapped with a newline character.)

## Value

A list of segmented texts, with each element of the list corresponding to one of the original texts.

## Examples

```
# same as tokenize()
identical(tokenize(uk2010immig, lower=FALSE), segment(uk2010immig, lower=FALSE))

# segment into paragraphs
segment(uk2010immig[3:4], "paragraphs")

# segment a text into sentences
segmentedChar <- segment(uk2010immig, "sentences")
segmentedChar[2]
# segment a corpus into sentences
segmentedCorpus <- segment(corpus(uk2010immig), "sentences")
identical(segmentedCorpus, segmentedChar)
```

---

settings

*Get or set the corpus settings*

---

## Description

Get or set the corpus settings

Get or set various settings in the corpus for the treatment of texts, such as rules for stemming, stop-words, collocations, etc. `settings(corp)` query the corps settings `settings(corp, settingname) <-` update the corpus settings

Get the settings from a which a [dfm](#) was created

## Usage

```
settings(x, ...)

## S3 method for class corpus
settings(corp, fields = NULL)

settings(corp, fields) <- value

## S3 method for class dfm
settings(x)
```

## Arguments

<code>x</code>	dfm from which settings are queried
<code>corp</code>	Corpus from/to which settings are queried or applied
<code>fields</code>	a valid corpus setting field name

## Examples

```
settings(tempcorpus, "stopwords")
tempdfm <- dfm(inaugCorpus)
tempdfmSW <- dfm(inaugCorpus, stopwords=TRUE)
settings(inaugCorpus, "stopwords") <- TRUE
tempdfmSW <- dfm(inaugCorpus)
tempdfm <- dfm(inaugCorpus, stem=TRUE)
settings(tempdfm)
```

---

sort.dfm	<i>sort a dfm by one or more margins</i>
----------	--

---

## Description

Sorts a [dfm](#) by frequency of total features, total features in documents, or both

## Usage

```
## S3 method for class dfm
sort(x, decreasing = TRUE, margin = c("features", "docs",
  "both"))
```

## Arguments

decreasing	TRUE (default) if sort will be in descending order, otherwise sort in increasing order
margin	which margin to sort on features to sort by frequency of features, docs to sort by total feature counts in documents, and both to sort by both
dfm	Document-feature matrix created by <a href="#">dfm</a>

## Value

A sorted [dfm](#) matrix object

## Author(s)

Ken Benoit

## Examples

```
dtm <- dfm(inaugCorpus)
dtm[1:10, 1:5]
dtm <- sort(dtm)
sort(dtm)[1:10, 1:5]
sort(dtm, TRUE, "both")[1:10, 1:5] # note that the decreasing=TRUE argument
# must be second, because of the order of the
# formals in the generic method of sort()
```

---

stopwords	<i>A named list containing common stopwords in 14 languages</i>
-----------	---

---

## Description

SMART English stopwords from the SMART information retrieval system (obtained from <http://jmlr.csail.mit.edu/papers/smart-stop-list/english.stop>) and a set of stopword lists from the Snowball stemmer project in different languages (obtained from [http://svn.tartarus.org/snowball/trunk/website/algorithms/\\*/stop.txt](http://svn.tartarus.org/snowball/trunk/website/algorithms/*/stop.txt)). Supported languages are danish, dutch, english, finnish, french, german, hungarian, italian, norwegian, portuguese, russian, spanish, and swedish. Language names are case sensitive. Alternatively, their IETF language tags may be used.

---

stopwordsGet	<i>access stopwords</i>
--------------	-------------------------

---

### Description

This function retrieves stopwords from the type specified in the `kind` argument and returns the stopword list as a character vector. The default is English. See [stopwords](#) for information about the list.

### Usage

```
stopwordsGet(kind = "english")
```

### Arguments

<code>kind</code>	The pre-set kind of stopwords (as a character string)
-------------------	---

### Value

a character vector or dfm with stopwords removed

### Examples

```
stopwordsGet()
stopwordsGet("italian")
```

---

stopwordsRemove	<i>remove stopwords from a text or dfm</i>
-----------------	--

---

### Description

This function takes a character vector or [dfm](#) and removes words in the remove common or 'semantically empty' words from a text. See [stopwordsGet](#) for the information about the default lists.

### Usage

```
stopwordsRemove(text, stopwords = NULL)

## S3 method for class character
stopwordsRemove(text, stopwords = NULL)

## S3 method for class dfm
stopwordsRemove(text, stopwords = NULL)
```

### Arguments

<code>text</code>	Text from which stopwords will be removed
<code>stopwords</code>	Character vector of stopwords to remove - if none is supplied, a default set of English stopwords is used

## Details

This function takes a character vector 'text' and removes words in the list provided in stopwords. If no list of stopwords is provided a default list for English is used. The function [stopwordsGet](#) can load a default set of stopwords for many languages.

## Value

a character vector or dfm with stopwords removed

## Examples

```
## examples for character objects
someText <- "Here is an example of text containing some stopwords we want to remove."
itText <- "Ecco un esempio di testo contenente alcune parole non significative che vogliamo rimuovere."
stopwordsRemove(someText)
stopwordsRemove(someText, stopwordsGet("SMART"))
stopwordsRemove(itText, stopwordsGet("italian"))
stopwordsRemove(someText, c("containing", "example"))

## example for dfm objects
docmat <- dfm(uk2010immig)
docmatNostopwords <- stopwordsRemove(docmat)
dim(docmat)
dim(docmatNostopwords)
dim(stopwordsRemove(docmat, stopwordsGet("SMART")))
```

---

subset.corpus	<i>extract a subset of a corpus</i>
---------------	-------------------------------------

---

## Description

Works just like the normal subset command but for corpus objects

## Usage

```
## S3 method for class corpus
subset(corpus, subset = NULL, select = NULL)
```

## Arguments

corpus	corpus object to be subsetted.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	expression, indicating the attributes to select from the corpus

## Value

corpus object

**Examples**

```
## Not run:
data(inaugCorpus)
summary(subset(inaugCorpus, Year>1980))

## End(Not run)
```

---

summary.corpus	<i>Corpus summary</i>
----------------	-----------------------

---

**Description**

Displays information about a corpus object, including attributes and metadata such as date of number of texts, creation and source.

**Usage**

```
## S3 method for class corpus
summary(corp, n = 100, verbose = TRUE, showmeta = FALSE)
```

**Arguments**

corp	corpus to be summarized
n	maximum number of texts to describe, default=100
verbose	FALSE to turn off printed output
showmeta	TRUE to include document-level meta-data

**Examples**

```
summary(inaugCorpus)
summary(inaugCorpus, n=10)
mycorpus <- corpus(uk2010immig, docvars=data.frame(party=names(uk2010immig)), enc="UTF-8")
summary(mycorpus, showmeta=TRUE) # show the meta-data
mysummary <- summary(mycorpus, verbose=FALSE) # (quietly) assign the results
mysummary$Types / mysummary$Tokens          # crude type-token ratio
```

---

syllableCounts	<i>A named list mapping words to counts of their syllables</i>
----------------	--

---

**Description**

A named list mapping words to counts of their syllables, generated from the CMU pronunciation dictionary

**References**

<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>

**Examples**

```
data(syllableCounts)
syllableCounts["sixths"]
syllableCounts["onomatopeia"]
```

---

texts	<i>get or set corpus texts</i>
-------	--------------------------------

---

**Description**

Get or replace the texts in a quanteda corpus object.

**Usage**

```
texts(corp)

texts(corp) <- value
```

**Arguments**

corp	A quanteda corpus object
rownames	If TRUE, overwrite the names of the documents with names from assigned object.

**Value**

For texts, a character vector of the texts in the corpus.

For texts <-, the corpus with the updated texts.

**Examples**

```
texts(inaugCorpus)[1]
sapply(texts(inaugCorpus), nchar) # length in characters of the inaugural corpus texts

## this doesnt work yet - need to overload [ for this replacement function
# texts(inaugTexts)[55] <- "GW Bushs second inaugural address, the condensed version."
```

---

tf	<i>normalizes the term frequencies a dfm</i>
----	--

---

**Description**

Returns a matrix of term weights, as a [dfm](#) object

**Usage**

```
tf(x)
```

**Arguments**

dfm                      Document-feature matrix created by [dfm](#)

**Value**

A dfm matrix object where values are relative term proportions within the document

**Author(s)**

Ken Benoit

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tf(dtm)[1:10, 100:110]
```

---

tfidf.dfm	<i>compute the tf-idf weights of a dfm</i>
-----------	--

---

**Description**

Returns a matrix of tf-idf weights, as a [dfm](#) object

**Usage**

```
## S3 method for class dfm
tfidf(x, normalize = TRUE)
```

**Arguments**

x                      document-feature matrix created by [dfm](#)  
normalize              whether to normalize term frequency by document totals

**Value**

A dfm matrix object where values are tf-idf weights

**Author(s)**

Ken Benoit

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tfidf(dtm)[1:10, 100:110]
tfidf(dtm, normalize=FALSE)[1:10, 100:110]
```



---

tokenize	<i>tokenize a set of texts</i>
----------	--------------------------------

---

## Description

Tokenize the texts from a character vector or from a corpus.

## Usage

```
tokenize(x, ...)

## S3 method for class character
tokenize(x, simplify = FALSE, sep = " ", ...)

## S3 method for class corpus
tokenize(corpus, ...)
```

## Arguments

<code>x</code>	The text(s) or corpus to be tokenized
<code>...</code>	additional arguments passed to <a href="#">clean</a>
<code>simplify</code>	If TRUE, return a character vector of tokens rather than a list of length <a href="#">ndoc</a> (texts), with each element of the list containing a character vector of the tokens corresponding to that text.

## Value

A list of length [ndoc](#)(x) of the tokens found in each text.

A list of length [ndoc](#)(texts) of the tokens found in each text.

## Examples

```
# same for character vectors and for lists
tokensFromChar <- tokenize(inaugTexts)
tokensFromCorp <- tokenize(inaugCorpus)
identical(tokensFromChar, tokensFromCorp)
str(tokensFromChar)
# returned as a list
head(tokenize(inaugTexts[57])[[1]], 10)
# returned as a character vector using simplify=TRUE
head(tokenize(inaugTexts[57], simplify=TRUE), 10)

# demonstrate some options with clean
head(tokenize(inaugTexts[57], simplify=TRUE, lower=FALSE), 30)
```

---

topfeatures	<i>list the most frequent features</i>
-------------	--

---

### Description

List the most frequently occurring features.

### Usage

```
topfeatures(x, n = 10, decreasing = TRUE)
```

```
## S3 method for class dfm
topfeatures(x, n = 10, decreasing = TRUE)
```

### Value

A named numeric vector of feature counts, where the names are the feature labels.

### Examples

```
topfeatures(dfm(inaugCorpus))
topfeatures(dfm(inaugCorpus, stopwords=TRUE))
# least frequent features
topfeatures(dfm(inaugCorpus), decreasing=FALSE)
```

---

trimdfm	<i>Trim a dfm based on a subset of features and words</i>
---------	---

---

### Description

Returns a document by feature matrix reduced in size based on document and term frequency, and/or subsampling.

### Usage

```
trimdfm(x, minCount = 5, minDoc = 5, sample = NULL, verbose = TRUE)
```

### Arguments

x	document-feature matrix created by <a href="#">dfm</a>
minCount	minimum feature count
minDoc	minimum number of documents in which a feature appears
sample	how many features to retain (based on random selection)
verbose	print messages

### Value

A [dfm](#) object reduced in size.

**Author(s)**

Ken Benoit adapted from code by Will Lowe (see [trim](#))

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dim(dtm)
dtmReduced <- trimdfm(dtm, minCount=10, minDoc=2) # only words occurring at least 5 times and in at least 2
dim(dtmReduced)
dtmSampled <- trimdfm(dtm, sample=200) # top 200 words
dim(dtmSampled) # 196 x 200 words
```

---

twitterSearch	<i>work-in-progress from-scratch interface to Twitter search API</i>
---------------	--

---

**Description**

work-in-progress from-scratch interface to Twitter search API

**Usage**

```
twitterSearch()
```

---

twitterStreamer	<i>work-in-progress interface to Twitter streaming API</i>
-----------------	--

---

**Description**

work-in-progress interface to Twitter streaming API

**Usage**

```
twitterStreamer()
```

---

twitterTerms	<i>make a corpus object from results of a twitter REST search</i>
--------------	---

---

### Description

All of the attributes returned by the twitterR library call are included as attributes in the corpus. A oauth key is required, for further instruction about the oauth process see: <https://dev.twitter.com/apps/new> and the twitterR documentation

### Usage

```
twitterTerms(query, numResults = 50, key, cons_secret, token, access_secret)
```

### Arguments

query	Search string for twitter
numResults	Number of results desired.
key	Number of results desired.
cons_secret	'your consumer secret here'
token	'your access token here'
access_secret	'your access secret here'
key	'your consumer key here'

### Examples

```
## Not run:
twCorp <- twitterTerms(example, 10, key, cons_secret, token, access_secret)

## End(Not run)
```

---

uk2010immig	<i>Immigration-related sections of 2010 UK party manifestos</i>
-------------	---

---

### Description

Extracts from the election manifestos of 9 UK political parties from 2010, related to immigration or asylum-seekers.

### Format

A named character vector of plain ASCII texts

### Examples

```
data(uk2010immig)
uk2010immigCorpus <- corpus(uk2010immig, docvars=list(party=names(uk2010immig)))
language(uk2010immigCorpus) <- "english"
encoding(uk2010immigCorpus) <- "UTF-8"
summary(uk2010immigCorpus)
```

# Index

`_language`, 6  
`as.DocumentTermMatrix`, 10  
  
bigrams, 2  
  
clean, 3, 25, 33  
collocations, 4  
corpus, 5  
countSyllables, 6  
  
describeTexts, 7  
dfm, 6, 8, 8, 9–11, 13, 22, 23, 26–28, 31, 32, 34  
dfm2ldaformat, 9  
dfm2tmformat, 10  
directory, 5, 11  
docnames, 11  
docnames<- (docnames), 11  
documents, 11  
DocumentTermMatrix, 10  
docvar, 5  
docvars, 6, 12  
docvars<- (docvars), 12  
  
encoding, 6  
  
features (features.dfm), 13  
features.dfm, 13  
file, 11  
filenames, 5  
flatten.dictionary, 13  
  
getRootFileNames, 14  
getTextDir, 15  
getTextDirGui, 15  
getTextFiles, 16  
  
inaugCorpus, 17  
inaugTexts (inaugCorpus), 17  
is.corpus (corpus), 5  
is.dfm (dfm), 8  
  
kwic, 17  
  
language, 6, 18  
  
lda, 9  
lda.collapsed.gibbs.sampler, 9  
likelihood.test, 19  
  
metacorporus, 6, 19  
metacorporus<- (metacorporus), 19  
metadoc, 6, 20  
  
ndoc, 21, 33  
ngram, 3  
ngrams, 21  
  
plot.dfm, 22  
preprocess, 23  
  
quanteda, 17, 24  
quanteda-package (quanteda), 24  
  
readWStatDict, 24  
regex, 25  
  
segment, 25  
settings, 6, 23, 26  
settings<- (settings), 26  
simple triplet matrix, 10  
sort.dfm, 27  
stopwords, 8, 27, 28  
stopwordsGet, 28, 28, 29  
stopwordsRemove, 28  
strheight, 22  
strwidth, 22  
subset.corpus, 29  
summary.corpus, 30  
syllableCounts, 30  
  
text, 22  
texts, 6, 31  
texts<- (texts), 31  
tf, 31  
tfidf.dfm, 32  
tokenise (tokenize), 33  
tokenize, 3, 21, 33  
topfeatures, 34  
trim, 35  
trimdfm, 34

twitterSearch, [35](#)  
twitterStreamer, [35](#)  
twitterTerms, [36](#)

uk2010immig, [36](#)  
unlist, [19](#)

wfm, [9](#)  
wordcloud, [22](#)